UNIVERSIDAD: niversidad Nacional de La Plata.

NUCLEO DISCIPLINARIO/COMITÉ ACADEMICO/OTROS TEMAS: Redes Académicas

TITULO DEL TRABAJO: **HERRAMIENTA DE REALIDAD VIRTUAL PARA DISEÑO DE INTERFACES GRAFICAS 3D EN JAVA.**

AUTOR(ES): R. Javier Vegas, Federico Cristina, Sebastián H. Dapoto, Verónica Artola

DIRECCIÓN: Dr.Marcelo Naiouf, Mg. Claudia Russo

CORREOS ELECTRÓNICOS DE LOS AUTORES: {jvegas, fcristina, sdapoto, vartola}@lidi.info.unlp.edu.ar

PALABRAS CLAVES: Realidad Virtual, Ambiente 3D, Java 3D

INTRODUCCIÓN

El trabajo consiste en desarrollar un plug-in para la plataforma Eclipse de desarrollo en Java que alternativamente puede utilizarse como herramienta de dicho lenguaje independientemente de esa plataforma. La herramienta debe permitir diseñar ambientes 3D para ser utilizados como interfaz gráfica de otras aplicaciones desarrolladas en Java o alternativamente exportadas para otros usos en formatos conocidos como VRML [11] y 3DStudio. Todo esto valiéndose del uso de la Realidad Virtual.

Como prueba de aplicación práctica de la herramienta, se desarrollará un prototipo de aplicación gráfica para la reconstrucción de objetos a partir de imágenes de básica funcionalidad.

OBJETIVOS

Se desea obtener una biblioteca o API para ser reutilizada en otras aplicaciones Java como es el caso de AWT o Swing para interfaces gráficas 2D. A partir de esto, la interfaz gráfica 3D deseada puede obtenerse interactuando con el plug-in desde Eclipse o alternativamente utilizarlo como una aplicación Java Stand Alone o también debe ser posible construirla por medio de código programado en alguna clase de Java.

Finalmente son objetivos de este trabajo, analizar la posibilidad de interfaz entre el producto a desarrollar y el ambiente de trabajo Matlab [4] [5] [6] y evaluar que modelo puede llegar a utilizarse para el diseño de la aplicación (P2P, Cliente/Servidor).

MATERIALES Y MÉTODOS

Estructura de la Aplicación

El sistema se basa en un framework Cliente/Servidor desarrollado en Java [1] y la reutilización de las clases de la API Java3D de Sun [2] [3].

Básicamente nos encontramos con una serie de clases en el thread principal que constituyen la parte central de la aplicación como es el escenario con sus controles, manejo de dispositivos, obstáculos, etc. Además existen dos hilos secundarios que se encargan de la comunicación con el server y en caso de ser necesario, la comunicación con un “cliente secundario” que se encarga de recibir la vista del cliente actual pero en forma parcial o con una vista 2D, lo que permite el uso del HMD con toda su funcionalidad (zoom out a 640 x 480 pixels y visión estereoscópica).

En cuanto al manejo de dispositivos tanto de entrada como de salida existen clases que permiten un manejo por capas. Entre ellas aparecen InputDevice que es una Interface de Java3D. Dicha interface debe ser implementada por todos los dispositivos que utilizaremos en el sistema y por eso se deben definir subclases de Glove, pPort, j3dTrackerInputDevice, etc; que son las clases que se encargan de estar con contacto con

los drivers del guante de datos, puerto paralelo y tracker respectivamente. Ante esto se decidió implementar una clase llamada InputDevices que se encargue del manejo de las mismas y que además funcione como Bridge entre estas clases y TouristControls que es quien recibe las ordenes del usuario y actúa en consecuencia. La misma idea se maneja para el tratamiento de colisiones entre objetos, comportamientos e iluminación en general para lo cual se dispone de una gran cantidad de clases predefinidas por Java3D.

Hardware utilizado

Los periféricos utilizados en el sistema van desde los estándares Mouse y Teclado hasta los específicos para aplicaciones de realidad virtual a saber:

- Mouse y Teclado
- Joystick analógico con retroalimentación
- Guante de datos P5 [7] [8] y Motor Vibrador para guante de datos [13]
- Head Mounted Display (o HMD) [9]
- Bicicleta fija adaptada para navegación en un NVE (Networked Virtual Envir.) [10]

En la figura 1 se muestra una forma de integrar los periféricos a la aplicación:

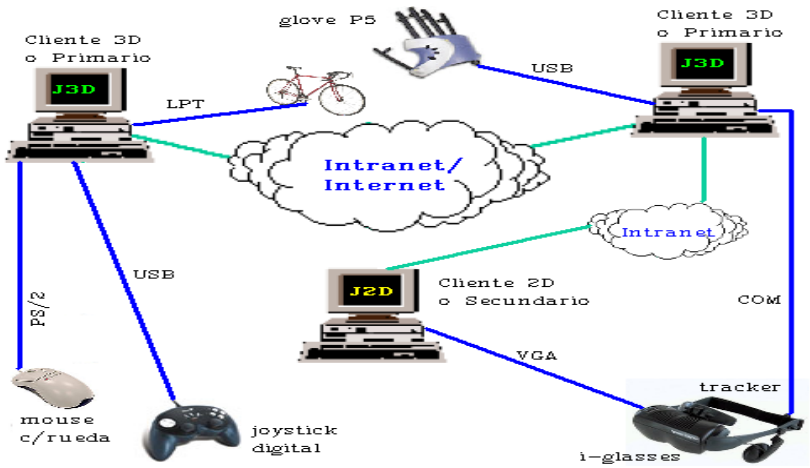

Figura 1 – Conexionado de Periféricos que interactuan con la aplicación

Existen ciertas restricciones de tiempo en el manejo de los periféricos lo cual le agrega un componente de Tiempo Real a la aplicación. Por tal motivo para cada periférico se tienen los diagramas de secuencia correspondientes como el siguiente que corresponde al polling del joystick

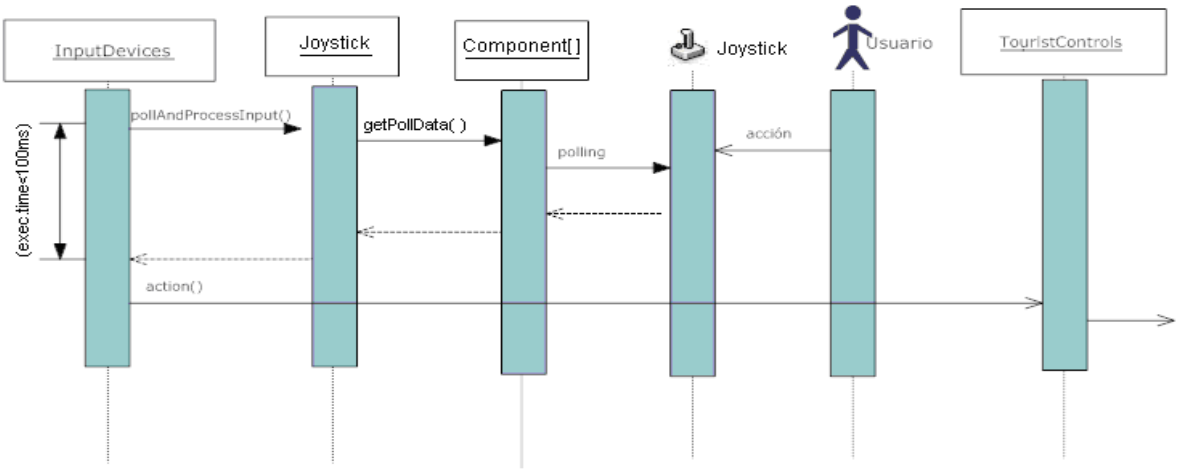

Figura 2 – Diagrama de Secuencia del proceso de polling del joystick

Comportamiento del modelo

Respecto al manejo de las colisiones entre objetos y comportamientos podemos decir que tanto la interacción como la animación son especificadas por medio de objetos de la clase Behavior. Un objeto Behavior cambia la escena en respuesta a eventos (teclas presionadas, movimientos del mouse, colisiones entre objetos, paso del tiempo, etc.)

Las llamadas “WakeupConditions” pueden ser:

- mouse, keyboard inputs
- colisiones
- cambio del tiempo y de frames
- movimiento de objetos
- movimiento de cámaras
- actividad de sensores (input devices)

Para la iluminación de las escenas en un Universo Virtual [2] [3] pueden usarse 4 tipos distintos de luces:

- Luz Ambiente
- Luz Direccional
- Luz Puntual
- Spotlights

El modelo C/S y el Escenario de Base

La idea básica, es que cada cliente mantiene una copia del modelo virtual que está navegando, además avisa los movimientos que realiza y se informa de los realizados por los otros usuarios por medio de un server.

En el siguiente esquema se ve el flujo de datos que corre por la red, junto con algunas clases que intervienen como así también información de puertos y otras estructuras utilizadas en la solución [1].

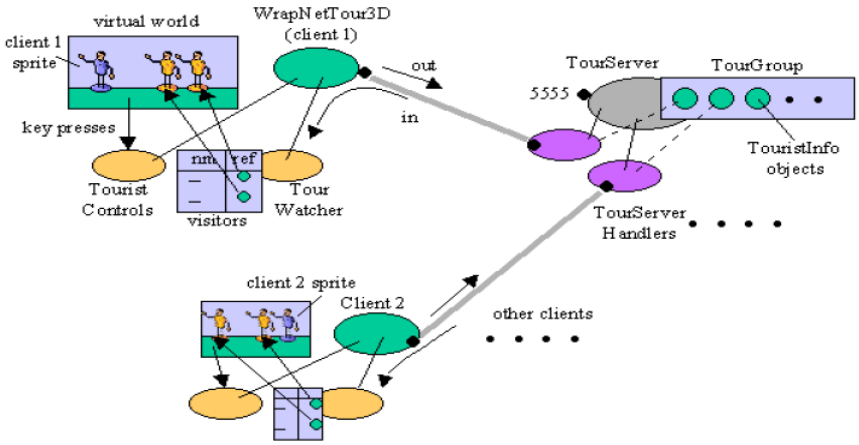

Figura 3 – Flujo de datos entre clientes y servidor

En el siguiente diagrama de clases pueden verse las partes del sistema que corresponden al Cliente [1]. Por motivos de legibilidad solo aparecen los nombres de las clases.

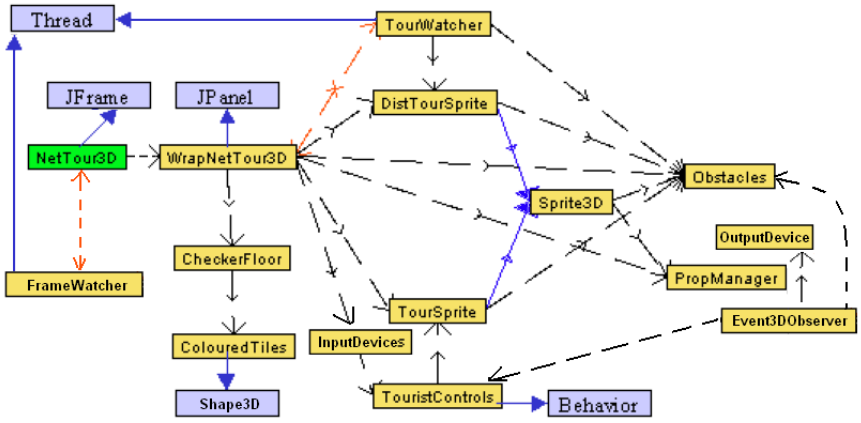

Figura 4 – Diagrama de las clases principales del lado Cliente

Como referencias tenemos que las clases con fondo de color lila son las provistas por el lenguaje. Las de fondo naranja son las creadas para la aplicación y la verde es la clase que actúa como punto de entrada a la aplicación en sí, del lado del cliente o servidor. Además las líneas punteadas indican “referencias a” y las azules sólidas indican extensiones de clases.

Básicamente nos encontramos con una serie de clases en el hilo principal que constituyen la parte central de la aplicación como es el escenario con sus controles, manejo de dispositivos, obstáculos, etc. Además existen dos hilos secundarios que se encargan de la comunicación con el server (TourWatcher) y en caso de ser necesario, la comunicación con un “cliente secundario” que se encarga de recibir la vista del cliente actual pero en forma

parcial o con una vista 2D, lo que permite el uso del HMD con toda su funcionalidad (zoom out a 640 x 480 pixels y visión estereoscópica), de esto se encarga la clase FrameWatcher.

En cuanto al manejo de dispositivos tanto de entrada como de salida solo aparecen las clases que están en las capas superiores.

Respecto del Cliente 2D o Cliente Secundario que ya fue mencionado, podemos decir que estará representado básicamente por una clase NetTour2D que se comunicará P2P con el Cliente 3D o Primario (NetTour3D) por medio de su correspondiente clase FrameWatcher para el intercambio de información gráfica ya procesada, dispuesta para su visualización en un HMD y que eventualmente se podría extender su funcionalidad de modo que este vinculo ya existente permita compartir otro tipo de recursos como dispositivos de entrada a distancia.

Como ejemplo de la información que se intercambia en la aplicación, en las figuras 5 y 6 se ven dos tipos de mensajes entre clientes y servidor. En la figura 7 se muestra el diagrama de actividades de uno de ellos[1].

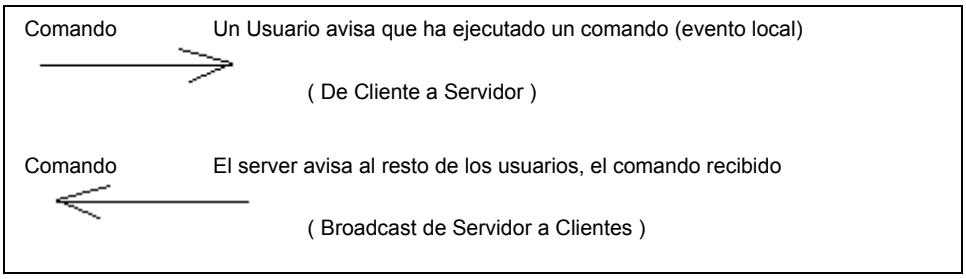

Figura 5 – Un usuario ejecuta un comando

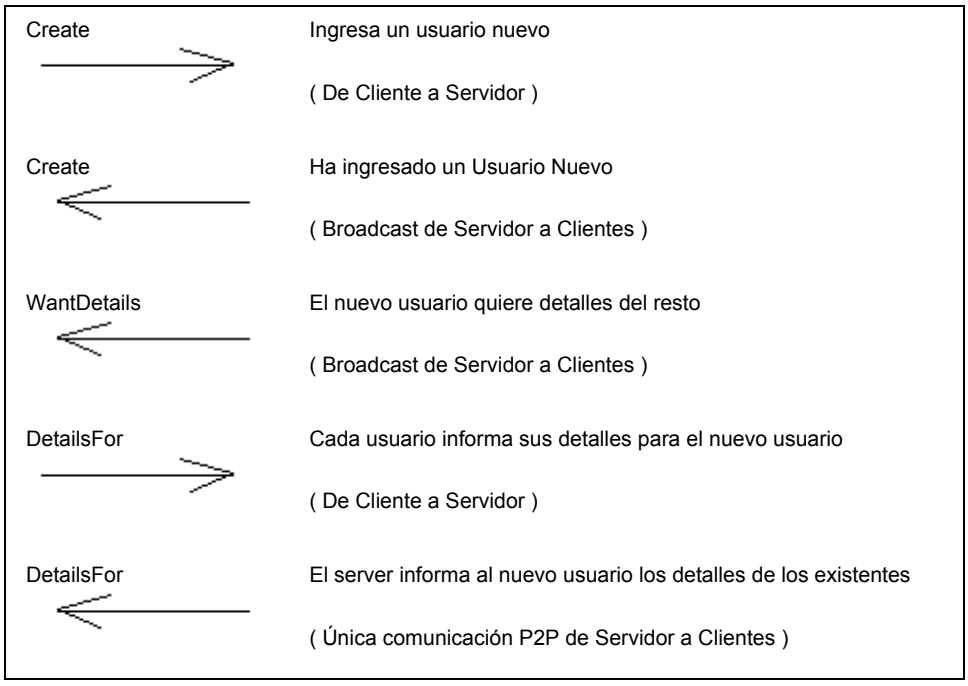

Figura 6 – Ingreso de un nuevo usuario al sistema

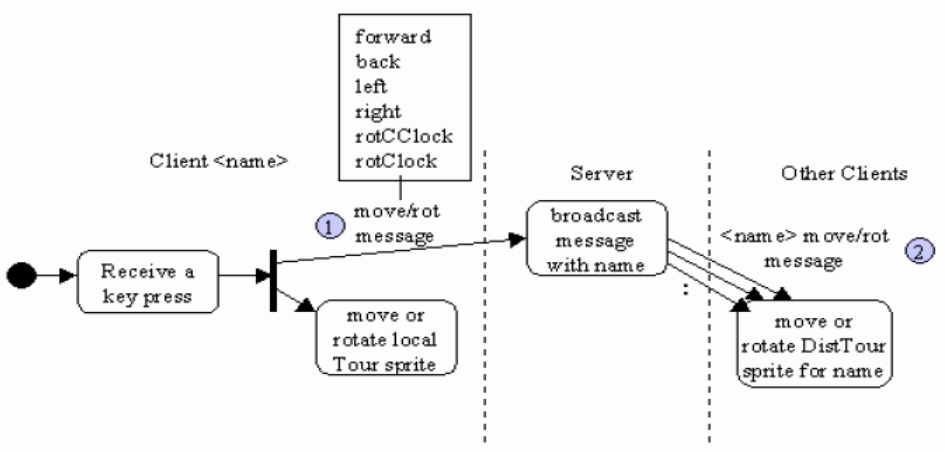

Figura 7 – Diagrama de Actividades de un Comando (movimiento o acción por parte de un usuario)

RESULTADOS Y DISCUSIÓN

Dentro de lo ya desarrollado se encuentra un prototipo de aplicación para la cual se dispone de una clase Java que corresponde a la GUI inicial de configuración de la aplicación colaborativa.y cuyo screenshot puede verse a continuación

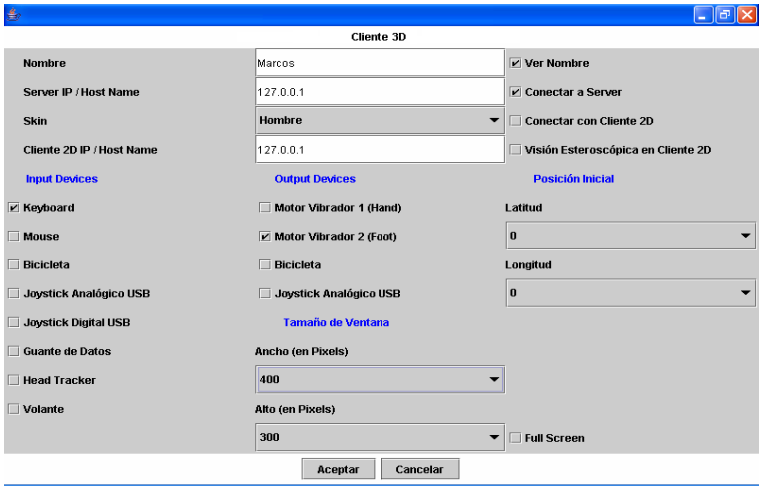

Figura 8 – Ventana Inicial del prototipo de la aplicación (lado cliente)

Cabe mencionar que el formato de los mensajes en la comunicación C/S que original proveía el framework, cambio gradualmente para permitir el reflejo en toda la red del skin usado por cada usuario y la visualización opcional de su nombre.

Respecto de los dispositivos de salida, podemos mencionar que como se indicó anteriormente, la creación de una clase OutputDevice y su jerarquia facilita el manejo de los periféricos de salida y los eventos que con ellos se desea asociar, en el modelo desarrollado encontramos 3 eventos distintos y 4 periféricos de salida. Por medio de esta clase es posible indicar para cada evento del entorno con que dispositivo de salida está relacionado. De esta forma por ejemplo el choque de un obstáculo se exterioriza por medio de cualquiera de los tres dispositivos asociados al evento que hayan sido habilitados desde la GUI inicial. Esta relación puede verse a continuación:

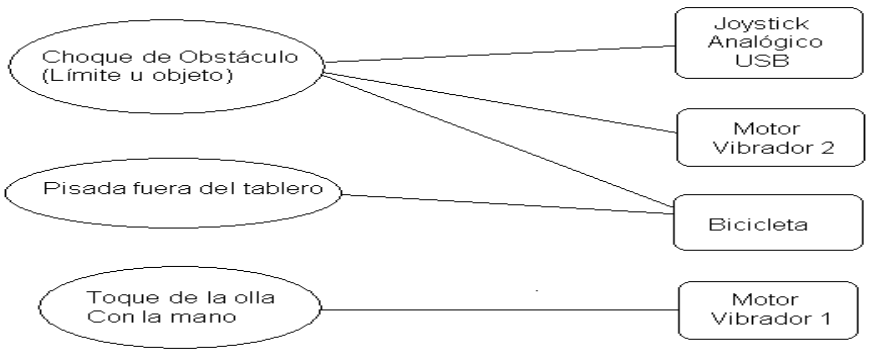

Figura 9 – Relaciones Evento – Periférico que maneja el prototipo de la aplicación

Además se presentan un en la figura 10, screenshots correspondiente a las pruebas realizadas con la aplicación corriendo en ella 2 clientes, los cuales interactúan entre si y con el NVE [10].

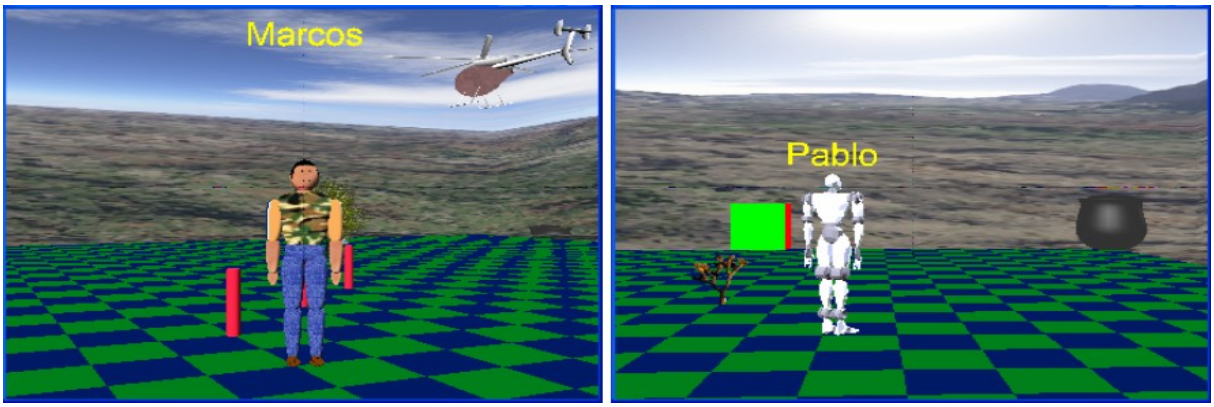

Figura 10 – Screenshots de la interacción entre usuarios

Por último se muestran pantallas de dos ambientes que fueron diseñados a partir del prototipo antes descrito y la necesidad de contar con ambientes 3D para otro tipo de proyectos como es el caso de la visualización de trayectorias (Figura 11) y la reconstrucción de objetos 3D [12] (Figura 12).

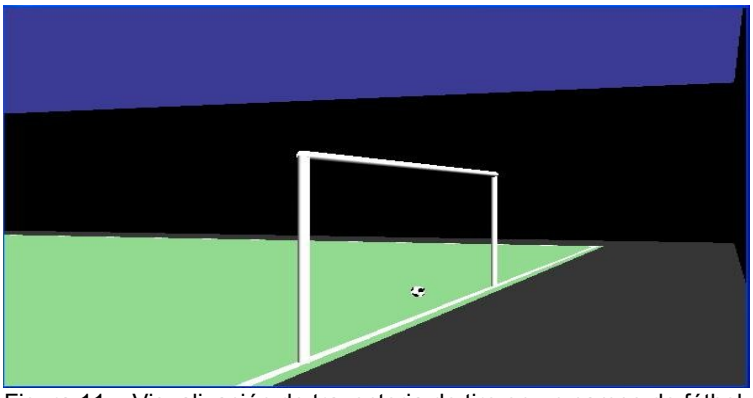

Figura 11 – Visualización de trayectoria de tiro en un campo de fútbol.

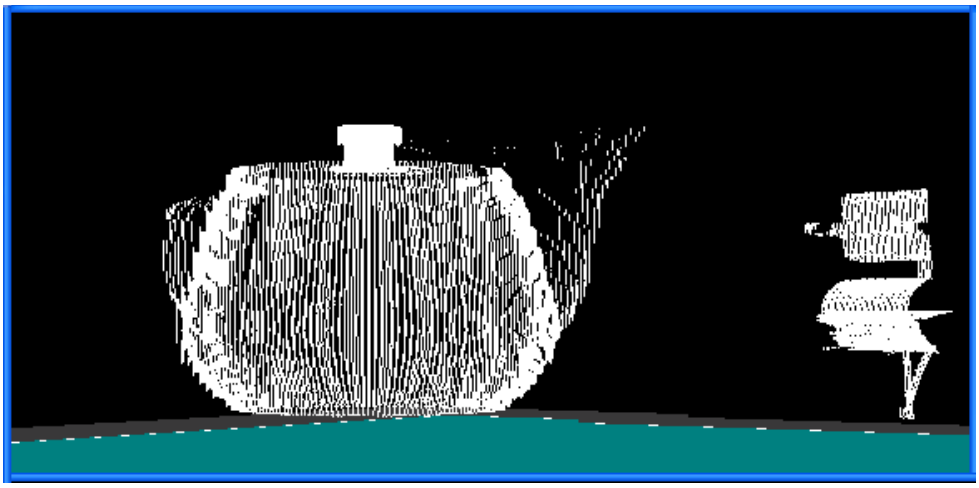

Figura 12 – Nube de puntos de una tetera y una silla en el proceso de reconstrucción 3D

La flexibilidad de la aplicación pudo testearse al llevar el ambiente inicialmente diseñado como un entorno de “fantasía” (fig.5) a otro adecuado para la visualización de trayectorias de tiros con poco esfuerzo (fig.7)

CONCLUSIONES

Se ha desarrollado parte de una herramienta que funcionando como un plug-in de Eclipse o como aplicación independiente permita diseñar entornos utilizables como interfaz gráfica 3D para otras aplicaciones en Java, o como ambientes navegables para otras aplicaciones como VRML; permitiendo además el manejo de eventos dentro del entorno generado y funcionalidades propias de un ambiente de Realidad Virtual. Además se dispone de una biblioteca con similar funcionalidad para ser reutilizada como API de Java, y un prototipo de aplicación gráfica 3D de reconstrucción de objetos basada en imágenes

REFERENCIAS

[1]Killer Game Programming in Java By Andrew Davison
http://fivedots.coe.psu.ac.th/~ad/jg/

[2] Java 3D™ API Specification
http://java.sun.com/products/java-media/3D/forDevelopers/j3dguide/j3dTOC.doc.html

[3] Java 3D API - 3D Geometry Compression
http://java.sun.com/products/java-media/3D/forDevelopers/j3dguide/AppendixCompress.doc.html

[4] MatLab – Functions By Category
http://www.mathworks.com/access/helpdesk/help/techdoc/ref/ref.html

[5] Calling MatLab from Java
http://www.cs.berkeley.edu/~kamin/matlab/JavaMatlab.html

[6] Calling Java from MatLab
http://math.carleton.ca/%7Ehelp/matlab/MathWorks_R13Doc/techdoc/matlab_external/ch_java.html
[7] Grupo Yahoo P5 Glove Community
http://groups.yahoo.com/group/p5glove/
[8] Sitios del guante de datos P5Glove
http://www.videogamealliance.com/VGA/video_game/P5.php
http://www.geocities.com/mellott124/p5.htm
http://www.alliancedistributors.com/Alliance_Brand/Products.php
http://www.vrealities.com/P5.html
[9] I-glasses Developer Page
http://iglasses.weirdoz.org/
[10] Virtual Environments
http://www.insead.fr/CALT/Encyclopedia/ComputerSciences/VR/vr.htm
http://en.wikipedia.org/wiki/Virtual_reality
[11] The Virtual Reality Modeling Language (VRML) and Java
http://web.nps.navy.mil/~brutzman/vrml/vrmljava
§[12] Voronoi-based Variational Reconstruction of Unoriented Point Sets. P. Alliez1 D. Cohen-Steiner1 Y. Tong2 M. Desbrun. INRIA Sophia-Antipolis Caltech
http://www.geometry.caltech.edu/pubs/ACTD07.pdf
[13] Especificación Webport Tunnel. Interface Ethernet/Serial.
http://www.sage.com.ar/Links/tunnel.htm